Fuzzy Classification using Grammatical Evolution for Structure Identification

Dominic A. Wilson	Devinder Kaur
Department of Electrical Engineering and Computer Science	*Department of Electrical Engineering and Computer Science*
University of Toledo	*University of Toledo*
Toledo, Ohio 43606, USA	*Toledo, Ohio 43606, USA*
dowilson@eng.utoledo.edu	dkaur@eecs.utoledo.edu

***Abstract* - The use of Grammatical Evolution for automating the structure identification of fuzzy rule based classification systems is investigated in this paper. To demonstrate its usefulness, we apply this method to the iris species identification problem. We achieve favorable results by evolving the rule base of a fuzzy system with fixed membership functions.**

I. Introduction

The acquisition of a fuzzy rulebase and the associated fuzzy set parameters from a set of input/output data is important in the development of fuzzy expert systems with or without the aid of human expertise. Several automated techniques have been proposed for the solution of this knowledge acquisition problem. The use of Grammatical Evolution provides the opportunity of constraining the format of the rule base to programs that are immediately executable on any machine that recognizes a specified grammar and of optimizing the rule base by genetic algorithm. In this paper this technique is applied to the iris species identification problem.

Grammatical Evolution (GE) is a method of evolving syntactically correct programs in an arbitrary context free language. GE has been tested with successful results on a symbolic regression problem, a trigonometric identity problem and a symbolic integration problem.

GE makes use of Genetic Algorithms as its method of finding solutions. In Genetic Algorithms use is made of selection, mutation and recombination operators to evolve the fittest genome for creating the program. Genetic Algorithms are introduced in the next section, prior to a discussion of the Grammatical Evolution, and the GE techniques used in this investigation. Subsequent sections detail the implementation and results and draw conclusions on the system identification.

II. Genetic Algorithms

Genetic Algorithms are forms of search procedures that use analogies of the genetic operations found in biology [1]. Darwin's theory of natural selection explains the survival of the fittest organisms in a population. The result is that genes encoding traits that favor fitness will be passed down through the generations, until they become common. Furthermore, when two organisms mate and produce offspring, characteristics of each parent are combined in new ways; so it is possible that the offspring will inherit features from both parents and be better suited to the environment Also rare genetic mutation will result in some individuals possessing traits that were not present in the population before.

In genetics, the basic structures are lengths of DNA called chromosomes. These are made up of a number of genes, each of which encodes a particular trait such as eye color; the range of 'values' that the gene can take are called the alleles. The particular combination of alleles on a chromosome can be thought of as a blueprint for an individual and is referred to as a genotype. The resulting physical characteristics of an organism upon expression of the genotype are known as the phenotype. The structures manipulated by a Genetic Algorithm are simplified model of these chromosomes that are usually made up of binary strings. The aim of Genetic Algorithm is to develop an optimal genotype by applying the simple genetic operators of selection, crossover and mutation. Genetic Algorithm fitness is usually measured as a function of how each phenotype measures up to a particular standard.

Selection is the process of determining which individuals will become parents for the next generation. The probability of an individual going on to breed is made some way proportional to its current fitness. This results in a 'mating pool' biased towards individuals of higher fitness. In order to simulate the genetic recombination that occurs naturally during sexual reproduction, some individuals will be used for 'breeding' via the crossover operator. A predefined crossover probability determines how often this operator is applied. In the single-point crossover used in this investigation, each parent string is divided into a 'head' and 'tail' section, and the tails are swapped over. For the mutation operation, each position in the binary string has a very small probability (typically 0.001) of having its value randomly altered.

III. Grammatical Evolution

Grammatical Evolution (GE) [2,3,4] is a method of evolving syntactically correct programs in an arbitrary context free language. The language to be generated is specified using a Backus Naur Form (BNF) grammar, consisting of a series of production rules mapping a set of non-terminal symbols to the set of terminals that are defined in the language.

Backus Naur Form (BNF) is a convention for expressing the grammar of a language in the form of production rules. The BNF grammar can be represented by a tuple, {N, T, P, S}, where N is the set of non-terminals, T is the set of terminals, P is the set of production rules that map the elements of N to T and S is the start symbol which is an element of N.

An example BNF grammar is displayed in. In the example BNF grammar, the symbol "::=" denotes that the non-terminal on the left of the production rule can be mapped into the symbol that appears on the right. The pipe symbol | is used to denote 'or'. Definitions can be recursive as the first production rule in shows.

IV. How Grammatical Evolution works

A linear genome made up of a variable number of 'genes' – each of which is represented by an 8-bit binary number – is used to control how the BNF grammar definition is mapped to an actual program. A Genetic Algorithm involving the use of selection, mutation and recombination is used to evolve the fittest genome for creating the program. The evolutionary aspect of GE is language independent, and can theoretically be used to generate functions of arbitrary complexity.

```
N = {statement, condition, function, variable, value}

T = {x,y,>,a,b,c,d,f1,f2,f3,f4,f5 }

S = <statement>

P =

<statement>     ::=IF <condition>  <statement> Else
          <statement> END;      (0)
                |  <function>   (1)
<condition>     ::= (<variable> > <value>)
<variable>      ::=x            (0)
                |  y            (1)
<value>         ::= a           (0)
                |  b            (1)
                |  c            (2)
                |  d            (3)
<function>      ::=f1           (0)
                |  f2           (1)
                |  f3           (2)
                |  f4           (3)
                |  f5           (4)
```

Fig. 1: Example BNF production rules

Fig. 2 outlines the mapping process used in both GE and biological systems. In this implementation, the binary string is segmented into 8 bit sections called codons. The codons are transformed to their corresponding integer values and these values used to decide which expansion to carry out whenever there is a choice between two or more possible expansions. A rule is selected by using the following [5]:

(Integer Codon Value) Mod (Number of production rules for current non-terminal)

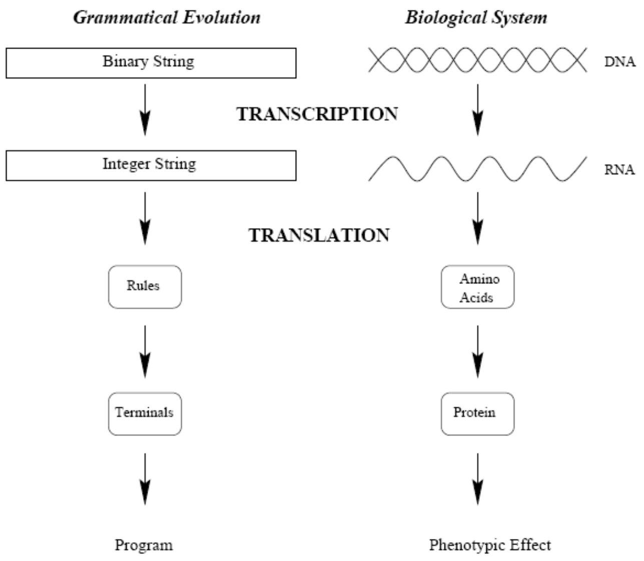

Fig. 2: Grammatical Evolution System and Biological System genotype to phenotype mapping [5]

Consider for example the following rule:

```
<value>   ::= a      (0)
        |  b    (1)
        |  c    (2)
        |  d    (3)
```

i.e. given the non-terminal "value" there are four production rules to select from. If we assume the integer corresponding to the codon being read is 29, then 29 MOD 4 = 1 resulting in the selection of "b" as the terminal. The system traverses the genome by reading a new codon each time a production rule has to be used.

104, 68, 44, 216, 17, 61, 123, 230,217, 71,250, 19, 62, 159, 122, 201, 123

Fig. 3: Example codon sequence

Consider a chromosome with the codon sequence expressed in Fig. 3 (expressed in decimal). The mapping process starts from the start symbol <statement>; because there is more than one production for this symbol (i.e. 2), we use the stated formula to get 104 MOD 2 = 0, and choose the production labeled (0). <statement> is replaced with:

IF <condition> <statement> Else <statement> END;

The next step involves the replacement of the leftmost non-terminal "<condition>" by "(<variable> > <value>)". Because there was only one production for "<condition>", a new codon value was not used. The current expression becomes:

```
IF (< variable> > <value>)
            <statement>
Else
            <statement>
END;
```

Calculating 68 MOD 2 = 0, specifies the use of the production: <variable>::=x.

Full resolution of the codon sequence yields the program:

```
IF(x>a)
   IF(y>b)
            (f1),
   ELSE
            (f2),
   END;,
ELSE
   IF(y>c)
            (f3),
   ELSE
            (f4),
   END;,
END;
```

The mapping system can be seen as having transformed a binary string into a program that partitions a feature space into a binary decision tree. The program gives the tree structure shown in . The fitness of the program is assessed by substituting feature values and measuring the difference between the calculated output and observed output of the process being identified.

V.GE FOR FUZZY CLASSIFICATION

An advantage of using GE in the context of evolving structures for fuzzy rulebases is the flexibility in defining different partitioning geometries based on a chosen grammar. The grammar also enforces the fuzzy reasoning method and if-then rules corresponding to fuzzy regions. Another important issue is the possibility of restricting exponential growth in the number of rules with an increase in the number of input parameters.

The GE based fuzzy classifier design involves: 1) Designing a grammar (in BNF form) that dictates the syntax of the partitioning and rule structure. 2) Determining a fitness function for using GE to optimize the classifier, and 3) Evolving the candidate structures until some predetermined stopping criteria is reached. The design objective is to efficiently search for compact classifiers that are highly interpretability have high classification accuracy.

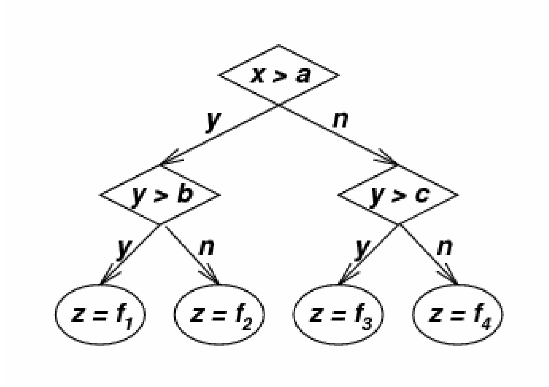

Fig. 4: Decision tree equivalent to program code.

The difference between using GE and other evolutionary approaches in the literature (see [6] for example) is that in GE the rule structure is evolvable within the constraint of the BNF. Also the transcription from bit string to evolved code uses redundancy to improve search efficiency.

```
Non_terminals = {<statement>,<ant>,<cons>}

Terminals = {Small, Small-Medium, Medium-Large, Large,
Virginica, Setosa, Versicolour}

(Start symbol) S = <statement>

(Production rules ) P=
<statement> ::=
If (SL is <ant>) and (SW is <ant>) and (PL is <ant>)and (PW is
<ant>) Then <cons>;
If (SL is <ant>) and (SW is <ant>) and (PL is <ant>)and (PW is
<ant>) Then <cons>;
.
. (repeat n times for n rules)
.
If (SL is <ant>) and (SW is <ant>) and (PL is <ant>)and (PW is
<ant>) Then <cons>;

<ant> ::=     Small              (0)
              | Small-Medium     (1)
              |Medium-Large      (2)
              |Large             (4)
              |All               (5)

<cons>::=      Virginica         (0)
              |Setosa            (1)
              |Versicolour       (2)
```

Fig. 5: Actual BNF used for Rule base

VI.IMPLEMENTATION

The data includes information about four features of the iris flowers: sepal length (SL), sepal width (SW), petal length (PL) and petal width (PW), which are used to classify a flower into one of the three species Setosa, Versicolor, and Virginica. There are 150 data items with 50 data items for each of the iris species.

Each input variable was divided into an equal number of fixed, triangular membership functions, as shown in Fig. 6. On experimentation with different numbers of symmetrically defined triangular antecedent membership function it was discovered that four functions per input variable gave the most consistent results. A membership function that encompasses all values of the domain was also used so that the system was able to generalize and leave out some of the inputs from the rules. This membership is denoted by "All" in the BNF.

Through the use of the grammar shown in Fig. 5, grammatical evolution was used to select and vary the antecedent clauses and the consequent set. Eight fuzzy rules per model were found adequate to give a consistent performance of above 97% classification successes. Fitness was measured by the number of data points that were correctly classified. Fig. 7 exemplifies the improvement in performance with generational change.

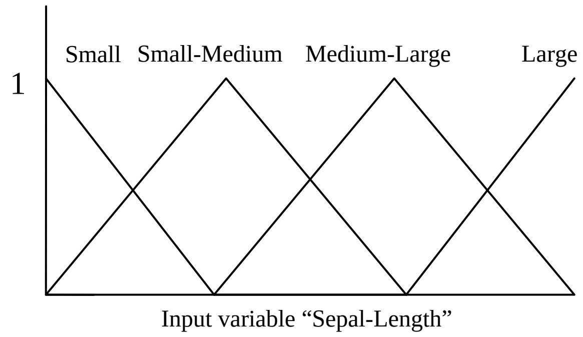

Fig. 6: Triangular membership functions for Sepal-Length

A fixed length binary string representation was used for the genomes. Five codons represented each rule, with the total number of rules being pre-specified by the genome length.

The genome population was randomly initialized and the following parameters were used for program runs:

Population size:	100
Number of generations:	50
Elite count:	2
Crossover rate:	0.8

Upon experimenting with different mutation rates and crossover types (see Table 1), single point crossover was found to be marginally more effective in final rule base performance. The result shown in Table 1 are the averages of ten runs. The crossover and mutation parameters were decided upon after various trials proved them to give relatively good performance. Table 2 shows a typical rule base that classifies 98% of the data correctly.

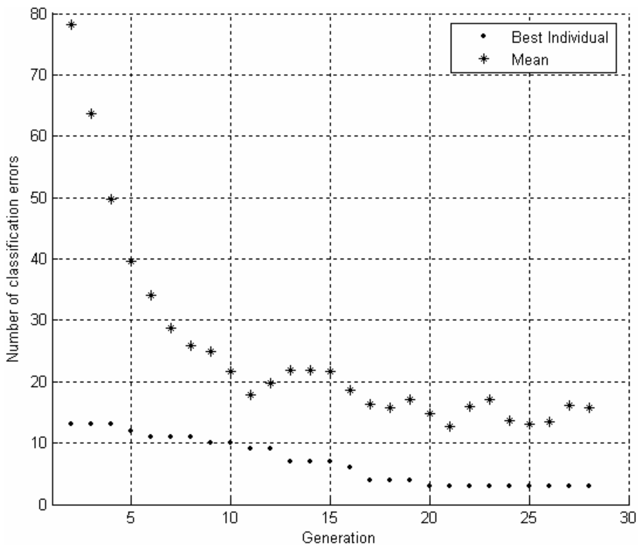

Fig. 7: improvement in performance with generation for a GE system that generated 8 fuzzy rules.

Mutation rate	single	double	scattered
0.005	97.8	97.3	97.5
0.01	97.6	97.2	97.7
0.015	97.7	97.3	97.3
0.02	97.5	97.6	97.8

Table 1: Classification rates for various mutation rates and crossover types

Rule#	Antecedent	Consequent
1	If (Sepal-Length is Large) and (Petal-Length is Large)	Virginica
2	If (Sepal-Length is Large) and (Petal-Length is Small-Medium)and (Petal-Width is Large)	Virginica
3	If (Sepal-Length is Small) and (Sepal-Width is Medium-Large) and (Petal-Length is Small)	Setosa
4	If (Petal-Length is Small)	Setosa
5	If (Sepal-Length is Medium-Large) and (Petal-Length is Large)and (Petal-Width is Small-Medium)	Virginica
6	If (Sepal-Length is Small-Medium) and (Sepal-Width is Small) and (Petal-Length is Medium -Large)and (Petal-Width is Large)	Virginica
7	If (Sepal-Length is Small) and (Sepal-Width is Small-Medium) and (Petal-Length is Large)and (Petal-Width is Medium-Large)	Virginica
8	If (Sepal-Width is Medium-Large) and (Petal-Length is Medium-Large)and (Petal-Width is Large)	Versicolour

Table 2: A Rulebase that gives 98% correct classification

VII. Conclusion

We have applied Grammatical evolution to the development and optimization of rule bases related to the iris species identification problem. This serves as a proof of concept of the possibility derive and optimize a fuzzy classification system using grammatical evolution. A success rate of more than 97% was obtained on average for some parameter settings when using a fixed set of membership functions. Future work will include the application of this technique to more complex classification problems and investigating ways of obtaining higher classification rates.

VIII. References

1 Goldberg, D.E. Genetic Algorithms in Search, Optimization and Machine Learning, pp. 1-14. Reading, MA: Addison Wesley, 1989.

2 Ryan, C., Collins, J.J., Micheal O'Neill, . Grammatical Evolution: Evolving Programs for an Arbitrary Language. Lecture Notes in Computer Science: Proceedings of the First European Workshop on Genetic Programming, pp. 83-95, 1998.

3 Ryan, C., Micheal O'Neill, Collins, J.J. Grammatical Evolution: Solving Trigonometric Identities. Proceedings of Mendel 1998: 4th International Mendel Conference on Genetic Algorithms, Optimisation Problems, Fuzzy Logic, Neural Networks, Rough Sets, pp. 111-119, 1998.

4 Ryan, C., Micheal O'Neill,.Grammatical Evolution: A Steady State Approach. Proceedings of the Second International Workshop on Frontiers in Evolutionary Algorithms, pp. 419-423, 1998.

5 Michael O'Neill, Conor Ryan, Grammatical evolution : evolutionary automatic programming in an arbitrary language, Kluwer Academic Publishers, 2003.

6 S.Y. Ho, H.M. Chen, S.J. Ho, T.K. Chen. Design of accurate classifiers with a compact fuzzy-rule base using an evolutionary scatter partition of feature space. IEEE Transactions on Systems, Man and Cybernetics, Part B 34:2 (2004) 1031-1044